

## Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals

Gray, Chaudhuri, Bosworth, Layman, Reichart, Venatrao, Fellow, Pirahesh  
DMKD 1997

Presented By: Kristen LeFevre

11/16/09

EECS 584

1

## Introduction

- Organizations want to analyze aggregate data to identify useful patterns, support business strategies
- Emphasis on complex, interactive, exploratory analysis of very large datasets
  - Contrast On-Line Analytic Processing (OLAP) with traditional On-Line Transaction Processing (OLTP)

11/16/09

EECS 584

2

## Complementary Trends

- Data Warehousing
  - Consolidate data from many sources into one repository
- OLAP
  - Complex SQL queries and views
  - Queries based on spreadsheet-style operations, aggregates, and "multidimensional" view of data
- Data Mining
  - Exploratory search for interesting trends and anomalies

11/16/09

EECS 584

3

## Multidimensional Data Model

- View aggregate values (e.g., counts and sums) at multiple levels of granularity
  - E.g., view car sales figures by make, model, color, date, and location

		Year			
		1992	1993	1994	
Make	Ford	40	25	22	87
	GM	65	21	20	106
	VW	22	31	18	71
		127	77	60	264

Make	Value
Taurus	60
Ranger	27
Buick	40
Saturn	66
Passat	21
Jetta	50

“Cross-Tabulation”

Group By

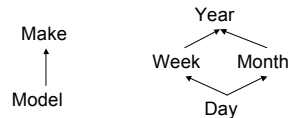
11/16/09

EECS 584

4

## Dimension Hierarchies

- Dimension attributes can be expressed at different levels of granularity



11/16/09

EECS 584

5

## OLAP Queries

- Influenced by SQL and by spreadsheets.
- A common operation is to aggregate a measure over one or more dimensions.
  - Find total sales.
  - Find total sales for each city, or for each state.
  - Find top five products ranked by total sales.
- Roll-up: Aggregating at different levels of a dimension hierarchy.
  - E.g., Given total sales by city, roll-up to get sales by state.

11/16/09

EECS 584

6

## OLAP Queries

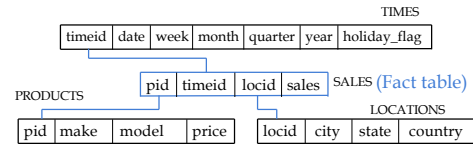
- **Drill-down:** The inverse of roll-up.
  - E.g., Given total sales by state, can drill-down to get total sales by city.
  - E.g., Can also drill-down on different dimension to get total sales by product for each state.
- **Pivoting:** Aggregation on selected dimensions to produce a cross-tabulation.

11/16/09

EECS 584

7

## Relational Representation



- Fact table in BCNF; dimension tables un-normalized.
- This kind of schema is very common in OLAP applications, and is called a **star schema**
  - Joining these relations is called a **star join**.

11/16/09

EECS 584

8

## Example -- Pivot Query

- Suppose we want to pivot on Year and Make
- Produces the following cross-tabulation

		Year			
		1992	1993	1994	
Make	Ford	40	25	22	87
	GM	65	21	20	106
	VW	22	31	18	71
		127	77	60	264

11/16/09

EECS 584

9

## Can we do this in SQL?

- Using a collection of SQL queries

```
SELECT SUM(S.sales)
FROM Sales S, Times T, Products P
WHERE S.timeid=T.timeid AND S.pid=P.pid
GROUP BY T.year, P.make
```

```
SELECT SUM(S.sales)
FROM Sales S, Times T
WHERE S.timeid=T.timeid
GROUP BY T.year
```

```
SELECT SUM(S.sales)
FROM Sales S, Products P
WHERE S.pid=P.pid
GROUP BY P.make
```

11/16/09

EECS 584

10

## The CUBE Operator

- Generalize the previous example
  - If there are  $k$  attributes, we have  $2^k$  possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions.
- **GROUP BY CUBE** pid, locid, timeid
  - Equivalent to grouping Sales on all eight subsets of the set {pid, locid, timeid}; each corresponds to an SQL query of the form:

```
SELECT SUM(S.sales)
FROM Sales S
GROUP BY grouping-list
```

11/16/09

EECS 584

11

## The CUBE Operator

- ALL -- A special value indicating omission of attribute from the aggregation
- E.g., consider aggregate SUM()
  - Total sum across all dimensions denoted ALL, ..., ALL, SUM(measure)

11/16/09

EECS 584

12

## CUBE Operator Example

Model	Year	Color	Sales
Chevy	1990	Red	25
Chevy	1990	White	8
Chevy	1991	Red	20
Chevy	1991	White	14
Ford	1990	Red	12
Ford	1990	White	22
Ford	1991	Red	7
Ford	1991	White	3

CUBE

Model	Year	Color	Sales
ALL	ALL	ALL	111
Chevy	ALL	ALL	67
Ford	ALL	ALL	44
ALL	1990	ALL	67
ALL	1991	ALL	44
ALL	ALL	Red	64
ALL	ALL	White	47
...	...	...	...
Chevy	1990	Red	25
Chevy	1990	White	8
Chevy	1991	Red	20
Chevy	1991	White	14
Ford	1990	Red	12
Ford	1990	White	22
Ford	1991	Red	7
Ford	1991	White	3

11/16/09 EECS 584 13

## ROLLUP Operator

- Another variation, good for rolling up dimensions
- GROUP BY ROLLUP year, month, date

Year	Month	Date	Sales
1995	Jan-1995	Jan-1-1995	5
1995	Jan-1995	Jan-2-1995	10
1995	Jan-1995	Jan-3-1995	3
1995	Feb-1995	Feb-1-1995	8
1995	Feb-1995	Feb-4-1995	7
2000	Jan-2000	Jan-6-2000	12

ROLLUP

Year	Month	Date	Sales
ALL	ALL	ALL	45
1995	ALL	ALL	33
1995	Jan-1995	ALL	18
1995	Jan-1995	Jan-1-1995	5
1995	Jan-1995	Jan-2-1995	10
1995	Jan-1995	Jan-3-1995	3
1995	Feb-1995	ALL	15
1995	Feb-1995	Feb-1-1995	8
1995	Feb-1995	Feb-4-1995	7
2000	ALL	ALL	12
2000	Jan-2000	ALL	12
2000	Jan-2000	Jan-6-2000	12

11/16/09 EECS 584 14

## How can we compute the CUBE?

- (Really) Naïve solution: Generate all  $2^k$  GROUP BY queries, and process each separately
- Better solution: If there are  $k$  attributes, use GROUP BY to compute the  $k$ -dimensional core
  - Compute super-aggregates from the core GROUP BY

11/16/09 EECS 584 15

## Example -- GROUP BY attributes form a lattice

11/16/09 EECS 584 16

## Complex and User-Defined Aggregates

- Approach won't work for all aggregate functions
- Aggregates have varying difficulty
  - Distributive:** can compute cube from next lower dimension values (e.g., count, min, max,...)
  - Algebraic:** can compute cube from next lower dimension + some additional information (e.g., average, variance, ...)
  - Holistic:** Need base data (e.g., median, ...)
- Distributive and Algebraic have simple and efficient algorithm: build higher dimensions from core
- Holistic computation seems to require multiple passes.
  - Can use samples to estimate them

11/16/09 EECS 584 17

## Materializing the CUBE

Harinarayan, Rajaraman, and Ullman 1996

- Suppose there are many queries using different aggregates
- Key optimization:** Pre-compute and store (aka *materialize*) portions of the data cube

**Query #1**  
SELECT SUM(Sales)  
...  
GROUP BY Year

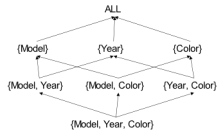
**Query #2**  
SELECT SUM(Sales)  
...  
GROUP BY Color

11/16/09 EECS 584 18

## Materializing the CUBE

### ■ Three Options:

1. Materialize nothing
2. Materialize whole cube
3. Materialize selected views



11/16/09

EECS 584

19

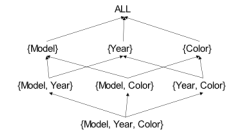
## Cost Optimization

### ■ Sources of Cost:

#### – Query Performance:

For query  $Q$ , # of rows in the table (view)  $Q_A$  used to answer  $Q$

#### – Storage Cost: Often constrained to store a fixed number of rows and/or views



Harinarayan et al. provide view-selection algorithms for several different versions of this problem

11/16/09

EECS 584

20

## Summary & Comments

- CUBE operator generalizes relational aggregates
- Useful in data warehousing / analysis / OLAP
- Much interesting work about how to efficiently compute the CUBE operator
- Later work about *materializing* the CUBE
  - Discussion Question: Without knowing the details of the Harinarayan et al. optimization framework, how would you decide which views to materialize? Are there other considerations?